POLYTECH SORBONNE

Sorbonne Université
2020-2021

Cours de Programmation en C – EI-2I

Travaux Pratiques

Objectif(s)

- ★ Rappels de C (types de données, opérateurs)
- ★ Entrées/Sorties
- ★ Boucles et traitement de file

Exercice 1 – Decodage RLE

1. Écrire un programme `decode.c` qui lit au clavier un entier et un caractère (séparés par un espace ou une tabulation), jusqu'à ce que l'entier soit -1.
2. Compléter ce programme pour que, à chaque couple entier `n`/caractère `c` lu, le programme :
   - affiche un saut de ligne si `n` vaut 0
   - affiche `n` fois le caractère `c` si $n \neq 0$

On utilisera les fichiers `code1.txt`, `code2.txt`, `code3.txt`, ainsi que la redirection de flux pour tester le programme (vérifier bien que votre programme tient compte de la syntaxe de ces fichiers tests).

Ce type d'encodage, bien que rudimentaire, est utilisé dans les fichiers images BMP et PCX, ainsi que sur certaines normes de communication par fax.

Exercice 2 – Codage RLE

Réaliser un programme `code.c` qui effectue l'opération inverse du décodage RLE de l'exercice précédent.
Le programme lit un à un des caractères au clavier, jusqu'au caractère `#`. Pour chaque caractère, le programme regarde s'il est identique aux suivants, et affiche le nombre d'occurrence de ce caractère et ce caractère.
Lorsqu'il s'agit du caractère de passage à la ligne, on affiche l'entier $\emptyset$ ainsi qu'un caractère quelconque. Lorsqu'il s'agit du caractère `#`, on affiche l'entier -1 et un caractère quelconque.
Par exemple, si on lit les caractères :

```
MMMM;;;U#
```

le programme affichera `4 M 3 ; 1 U -1 M`

Remarques :

- Il vous faudra pour cela commencer par écrire l'algorithme (en pseudo-code ou en français) avant de coder
- Vérifier que votre programme de décodage décode bien les textes codés par le programme de codage. Pour cela, modifier le programme de décodage pour qu'il marque le caractère de fin de fichier `#`
- Utiliser la redirection de flux (ou encore mieux un pipe `|`) pour enchaîner les deux programmes lors de vos essais

Exercice 3 – Indentation automatique

Le but de cet exercice est de réaliser un petit outil capable de mettre en forme un fichier source en C en gérant correctement l'indentation.

Listing 1: Un listing peu lisible...

```
#include <stdio.h>

int main(){
int i,j,n;
scanf("%d",&i);
if (i<0){
printf("Nombre negatif!!\n");
else{
n=1;
for(j=1;j<=i;j=j+1){
n=n*j;
}
printf("%d!=%dn",i,n);
}
return 0;
}
```

Listing 2: Le même, en plus lisible...

```
#include <stdio.h>

int main(){
    int i,j,n;
    scanf("%d",&i);
    if (i<0){
        printf("Nombre negatif!!\n");
    else{
        n=1;
        for(j=1;j<=i;j=j+1){
            n=n*j;
        }
        printf("%d!=%dn",i,n);
    }
    return 0;
}
```

Principe

Le principe de notre programme est le suivant : nous allons lire un texte caractère par caractère et le ré-écrire, au fur et à mesure. Le programme traitera le texte comme une file de caractères, qu'il lira un à un. On décidera ensuite si ce caractère doit être affiché dans le fichier destination, éventuellement précédé ou suivi d'espace(s), de tabulation(s) ou d'un passage à la ligne. Enfin, nous nous occuperons de quelques cas particuliers.
De plus, pour que le programme soit facile à utiliser, nous utiliserons le principe de **redirection de flux** pour gérer l'entrée et la sortie de notre progamme (la saisie et l'affichage se font sur les entrées/sorties standard, et c'est au moment de l'exécution que l'on décide de rediriger la sortie vers un fichier, et de lire depuis un fichier).

En outre, nous utiliserons la constante `EOF` (définie dans `stdio.h`) qui est la valeur renvoyée par `getchar` losrqu'il n'y a plus de caractères à lire.

Question 1

On veut, dans un premier temps, supprimer tous les espaces, tabulations et passages à la ligne lus et qui ne sont pas nécessaires. On se donne les règles suivantes :

- on ne reproduit pas les tabulations (`'\t'`) ni les espaces qui sont en début de ligne ;
- les sauts de ligne (`'\n'`) sont reproduits, mais s'il n'y en a plus que deux consécutifs (cela permet d'éviter les portions de code séparées par de trop grands espaces, et d'homogénéiser le code) ;

- dans tous les autres cas, le caractère est, bien sûr, reproduit.

Écrire le programme correspondant.

Question 2

On veut maintenant modifier ce programme pour produire l'indentation nécessaire : il faut rajouter le bon nombre de tabulations **après** chaque passage à la ligne (déterminé par le nombre d'accolades vues).
Il faut aussi afficher un message si le texte traité ne possède pas le même nombre d'accolades ouvrantes et fermantes.

Question 3

Maintenant que l'on dispose d'un programme générant l'indentation, il est possible d'appliquer le programme à son propre source. Malheureusement, il reste encore un tout petit problème : dans notre source, il y a des accolades ouvrantes et fermantes qui provoquent une tabulation alors qu'elles ne le devraient pas; il s'agit de { et } mis dans une chaîne de caractères (entre guillemets `"`) ou dans un caractère (entre `'`). Il faut donc analyser l'ouverture et la fermeture des guillemets `"` et `'` (*double quote* et *single quote*).
Il faut de plus analyser les commentaires pour que les apostrophes (`'`) s'y trouvant ne soient pas considérés comme des ouvertures de chaines de caractères.